ATID Application Development Framework Reference Manual – RFID900Mhz

Revision: Ver. 0.6

Date: January, 2014

ATID Co., Ltd.

Table of Contents

Acronym

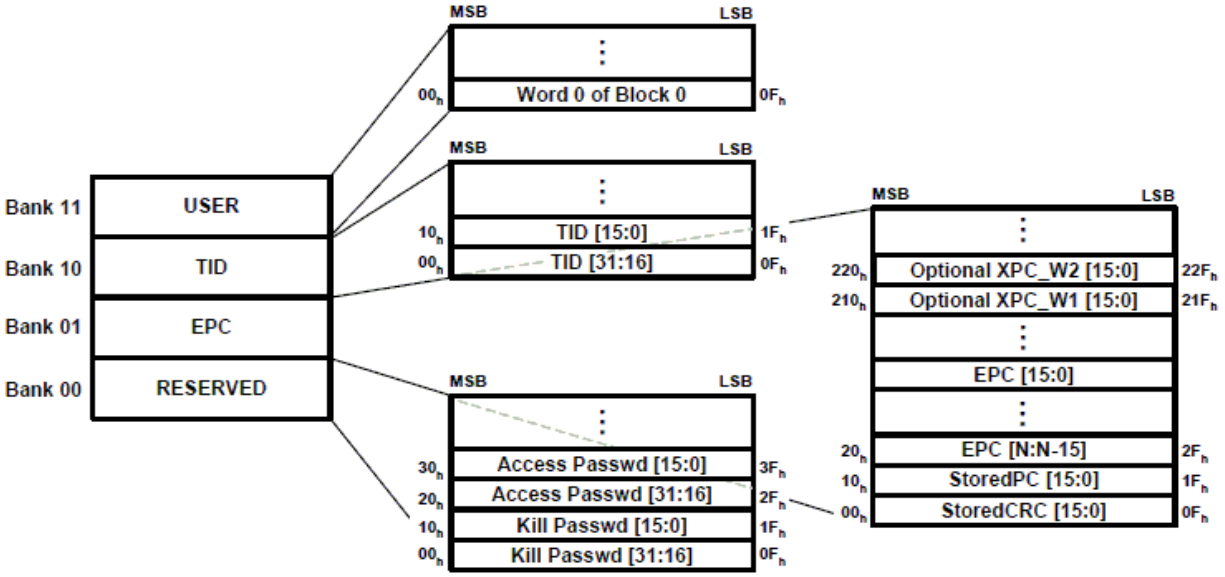
modules	descriptions
AADF	ATIDApplication Development Framework
EPC	Electronic Product Code

Revision History

Version	Date	Reason	Description	Author
0.1	2012/01/17	Draft		Y. J. CHO
0.2	2013/01/27	Update	- IsRunning added. - Stop(bool block) added. - GetModuleType, GetMaxPwr added. - Explanation of LBT Channel state complemented.	Y. J. CHO
0.3	2013/02/26	Update	- Register Control Function of R1000 added.	Y. J. CHO
0.4	2013/05/28	Update	- SetAirDura, TagFocus function added. - .NET API class constructor added. - typo correction.	Y. J. CHO
0.5	2013/09/12	Update	- change Max of PowerLevel to 30 - a description of initializing RFIDMASKPARAMS_EX MaskPattern members added. - the min, max Description of the members of AIRDURAPARAMS structure added.	Y. J. CHO
0.6	2014/01/10	Update	- ChannelState Function same as LBTChannelState function added. - Europe mode change function added..	Y. J. CHO

1 .NET API Reference

- ISO 18000 6C(EPC Class 1 Generation 2) Logical Memory Map



1.1 Enumerations

1.1.1 RFID_RESULT

The result of a call to functions.

- **RFID_RESULT_SUCCESS**
  Function executed successfully.
- **RFID_RESULT_OUTOFMEMORY**
  Failed to assign resource.
- **RFID_RESULT_INVALID_ARGS**
  Invalid parameter.
- **RFID_RESULT_FAILURE**
  Function execution failed.
- **RFID_RESULT_UNEXPECTED**
  Undefined Error.
- **RFID_RESULT_ALREADY_OPENED**
  RFID device port has already opened.
- **RFID_RESULT_INVALID_DEVICE**
  Do not support UHF reader type currently.
- **RFID_RESULT_NOT_CONNECTED**
  Module doesn't work properly, after be opened.
- **RFID_RESULT_NOT_DETECT**
  Cannot find Tag and command aborted.
- **RFID_RESULT_ACCESS_ERROR**
  Try to access with incorrect password, or try to access nonexistent region.
- **RFID_RESULT_NOT_OPENED**
  Call function without open.
- **RFID_RESULT_COMMAND_ERROR**
  Happened error while executing command, or received another command before finished one command.
- **RFID_RESULT_LOW_BATTERY**
  Module cannot work, because of low battery.
- **RFID_RESULT_UNKNOWN**
  UHF reader module happen Unknown Error.
- **RFID_RESULT_NOT_SUPPORTED**
  Not supported command currently.

- **RFID_RESULT_STOPPED**
  UHF reader module command aborted.
- **RFID_RESULT_POWER_OFF**
  UHF reader module power turned off.

1.1.2 MEM_BANK

Tag's Memory bank

- **RESERVED**
  memory where will be stored tag kill or access password.
- **EPC**
  memory where will be stored StoredCRC, EPC, XPC information.
- **TID**
  memory where will be stored ISO/IEC 15963 allocation class identifier, Tag serial number and so on.
- **USER**
  user specific data storage.

1.1.3 READ_TYPE

When reading Tag, can designate the type of tag (ISO-18000-6B or ISO-18000-6C[GEN2]) and then to read UID. And also can read UID which starting with a specific mask data tag.

- **EPC_GEN2_MULTI_TAG**
  Multiple read
- **EPC_GEN2_ONE_TAG**
  Single read

1.1.4 PERMALOCK_FIELD

Field where apply to Permalock.

- **ACCESS_PASSWORD**
  Tag's Access password field
- **KILL_PASSWORD**
  Tag's Kill password field
- **EPC**
  Tag's EPC bank
- **TID**
  Tag's TID bank

- **USER**
  Tag's USER bank

1.1.5 RFID_CALLBACK_TYPE

- **RFIDCALLBACKTYPE_DATA**
  because of module has already read memory data from tag, so it means that the Callback delegate has been executed.
- **RFIDCALLBACKTYPE_REPLY**
  because received the results of command execution from module, so it means that Callback delegate has been executed.

1.1.6 INVENTORIED_STATE

Inventoried state of tag which will be used as target while UHF module executing ReadEpc.

- **STATE_A**
  Inventoried state A
- **STATE_B**
  Inventoried state B
- **STATE_AB**
  Inventoried state A or B

1.1.7 SESSION_TYPE

Session which will be used while UHF module perform ReadEpc(Inventory).

- **SESSION_0**
  S0
- **SESSION_1**
  S1
- **SESSION_2**
  S2
- **SESSION_3**
  S3

1.1.8 SELECT_ACTION

Content that will apply to Target Flag of Tag

- **ACTION_0**
  000
- **ACTION_1**
  001
- **ACTION_2**
  010
- **ACTION_3**
  011
- **ACTION_4**
  100
- **ACTION_5**
  101
- **ACTION_6**
  110
- **ACTION_7**
  111

1.1.9 SELECT_TARGET

Set Target Flag of Tag which will be applied to the Select command.

- **S0**
  S0
- **S1**
  S1
- **S2**
  S2
- **S3**
  S3
- **SL**
  SL

You can change tag's state flag with below option.

	If SELECT_TARGET is SL		If SELECT_TARGET are between S0 ~ S3	
	Matching	Non-matching	Matching	Non-matching

ACTION_0	Assert SL	De-assert SL	→ A	→ B
ACTION_1	Assert SL	No Action	→ A	No Action
ACTION_2	No Action	De-assert SL	No Action	→ B
ACTION_3	Negate SL	No Action	Negate	No Action
ACTION_4	De-assert SL	Assert SL	→ B	→ A
ACTION_5	De-assert SL	No Action	→ B	No Action
ACTION_6	No Action	Assert SL	No Action	→ A
ACTION_7	No Action	Negate SL	No Action	Negate

1.2 Constants

1.2.1 MAX_MASKS_BYTES

Maximum byte which can be used as Making pattern

- **Public const int MAX_MASKS_BYTES 32**

1.2.2 MODULE_TYPE_NONE.

RFID Module is not equipped.

- **Public const int MODULE_TYPE_H1000 0x30**

1.2.3 MODULE_TYPE_H1000

January 2013 now, RFID module that corresponds to the country (excluding Japan),

Public const int MODULE_TYPE_H1000 0x31

1.2.4 MODULE_TYPE_I2000

January 2013 now, Japan only, RFID module that corresponds to Japan.

the changed Radio Law of JapanApplied

Public const int MODULE_TYPE_I2000 0x32

1.2.5 MODULE_TYPE_I900

January 2013 now, RFID Module that can be mounted on only AT980

Public const int MODULE_TYPE_I900 0x33

1.3 Reply Words List

If execute access tag function without using SyncMode (read one tag synchronically), registered Callback function will be executed, and if the CallbackTyps was UHFCALLBACKTYPE_REPLY, will be received one of the reply code as below:

- **"Other Error"**
  Other error.

- **"Memory Overrun"**
  Try to access inexistent memory region.

- **"Memory Locked"**
  Tag is locked.

- **"Insufficient Tag Power"**
  Cannot execute Write command because the tag power is insufficient.

- **"Non-specific Error"**
  Tag not supports (replay) code currently.

- **"Check Antenna"**
  Cannot connect Antenna

- **"Try after cooled"**
  Module is overheating.

- **"Insufficient PDA Power"**
  Module is overheating.

- **"Not Supported"**
  Unsupported function currently.

- **"Not Connected"**
  Module cannot work properly, after be opened.

- **"Not Opened"**
  Call function without open.

- **"Bad Access Password"**

  Try to access tag with incorrect Access Password.

- **"Invalid Parameter"**

  Transfer invalid parameter while call function.

- **"Command Error"**

  Happened error while executing command, or received another command before finished one command.

- **"Success"**

  Function executed successfully.

- **"Not Detect"**

  Command finished before detect Tag.

- **"Multi Read Stop"**

  Multi Read received a stop command from App, and to be stopped.

- **"EAS"**

  Detect EAS of NXP Tag.

1.4 Structures

1.4.1 LOCKUNLOCKFIELD

Store information that about lock or unlock each field of tag.
Lock or unlock only apply to set with True's member.

```
Public struct LOCKUNLOCKFIELD
{
   boolbAccessPassword;
   boolbEPC;
   boolbKillPassword;
   bool bTID;
   bool bUSER;
};
```

- **bAccessPassword**
  Tag's access password field
- **bEPC**
  Tag's EPC bank
- **bKillPassword**
  Tag's kill password field
- **bTID**
  Tag's TID bank
- **bUSER**
  Tag's USER bank.

Ex) in order to Access Password and Lock or Unlock EPC, to structure as below:

```
LOCKUNLOCKFIELD Lock = new LOCKUNLOCKFIELD();
Lock.bAccessPassword = true
Lock.bEPC = true
Lock.bKillPassword = false
Lock.bTID = false
Lock.bUSER = false
```

1.4.2 RFIDMASKPARAMS

While call the command which apply to the tag, only make those satisfied relevant condition tags respond.

```
Public struct RFIDMASKPARAMS
```

```
{
  MEM_BANK MemBank;
  uint nOffSet;
  String MaskPattern;
};
```

- **MemBank**
  Memory Bank of tag which will be used as Mask.
- **nOffSet**
  Nibble offset of selected memory bank.
- **MaskPattern**
  Mask pattern string which designated at 4bit.

Ex) when a tag's TID bank is "E2006004015F325A", and would like to use this TID bank as mask, the structure as below:

```
RFIDMASKPARAMS Mask = new RFIDMASKPARAMS();
Mask.MemBank = MEM_BANK.TID;
Mask.nOffSet = 1;
Mask.MaskPattern = "2006";
or
RFIDMASKPARAMS Mask = new RFIDMASKPARAMS();
Mask.MemBank = MEM_BANK.TID;
Mask.nOffSet = 11;
Mask.MaskPattern = "F325";
```

1.4.3 RFIDMASKPARAMS_EX

Using the function,while bit unit is masking pattern.

```
Public struct RFIDMASKPARAMS_EX
{
  MEM_BANK MemBank;
  uint nOffSet;
  uint nBits;
  byte[] MaskPattern;
};
```

- **MemBank**
  Tag's Memory Bank which will be used as Mask.
- **nOffSet**
  Selected Memory Bank's bit offset

- **nBits**
  bit of MaskPattern which will be used as masking pattern.
- **MaskPattern**
  ₩masking pattern byte array
  Initialize using MAX_MASK_BYTES constant.

1.4.4 RFIDSELMASKPARAMS_EX

Structures that composed with information of the Select command

```
Public struct RFIDSELMASKPARAMS_EX
{
  SELECT_ACTION ActionCode;
  uint Bits;
  byte[] MaskPattern;
  MEM_BANK MemBank;
  uint OffSet;
  SELECT_TARGET SelectTarget;
};
```

- **ActionCode**
  Content that will apply to Target Flag of Tag
- **Bits**
  bit of MaskPattern which will be used as masking pattern.
- **MaskPattern**
  bit array which will be used as masking pattern.
- **MemBank**
  Tag's Memory Bank which will be used as Mask.
- **OffSet**
  Selected Memory Bank's bit offset
- **SelectTarget**
  Set Target Flag of Tag which will be applied to the Select command.

1.4.5 RFIDCALLBACKDATA

Data which came from Callback delegate that registered in application program.
Using the data values, which came from while call Callback delegate, as parameters to call GetResult function.
The value while called GetResult function, can divide into Data or Reply according to CallbackType.

```
Public struct RFIDCALLBACKDATA
{
  RFID_CALLBACK_TYPE CallbackType;
  IntPtr lParam
  IntPtr wParam;
};
```

- **CallbackType**
  the executing reason of Callback delegate.(Data or Reply)
  Transmit byGetResult function's parameter.
- **lParam**
  Parameter 1 which in order to read value from module.
  Transmit byGetResult function's parameter.
- **wParam**
  Parameter 2 which in order to read value from module.
  Transmit byGetResult function's parameter.

1.4.6 AIRDURAPARAMS

Structure used to adjust the operating time of Tx Cycle module.

```
Public struct AIRDURAPARAMS
{
  uint nOffMs;
  uint nOnMs;
};
```

- **nOffMs**
  break time in Tx Cycle(millisecond)
  Min:0ms ~ Max:4000ms
- **nOnMs**
  operating time in Tx Cycle(millisecond)
  Min:0ms ~ Max:4000ms

1.5 Delegates

1.5.1 RfidCallbackProc

Callback delegate, that will process data while get reply from module.

Public delegate void **RfidCallbackProc**(RFIDCALLBACKDATA CallbackData);

- **CallbackData**

  UHFCALLBACKDATA object, which including data of getting from module.

1.6 Constructor

Initialize a new instance of RfidApi class.

public RfidApi()

1.7 Methods

1.7.1 PowerOn

Put voltage into the UHF RFID reader module.

RFID_RESULT PowerOn();

Parameters

None

Return Values

RFID_RESULT_SUCCESS will be returned if all the processes are performed normally, or already been Power On State.

RFID_RESULT_FAILURE will be returned if failed.

Notes

It will be called at the beginning of the Application.

1.7.2 PowerOff

Remove the power from UHF RFID reader module.

RFID_RESULT PowerOff();

Parameters

None

Return Values

RFID_RESULT_SUCCESS will be returned if all the processes are performed successfully,

1.7.3 Open

Create message queue, thread for receiving message from UHF system driver.

Open the port for communicate with UHF RFID reader module.

RFID_RESULT Open();

Parameters

None

Return Values

RFID_RESULT_SUCCESS will be returned if all the processes performed successfully.

RFID_RESULT_ALREADY_OPENED will be returned if the port has already opened, and be regarded as the port open successfully.

1.7.4 Close

Release the message queue, thread which created while Open(), and in order to communicate with UHF RFID reader module to close the port.

RFID_RESULT Close();

Parameters

None

Return Values

RFID_RESULT_SUCCESS will be returned if all the processes are performed successfully,

1.7.5 IsOpened

Checking whether the port was openwhich communicate with UHF RFID reader module.

BOOL IsOpened()

Parameters

None

Return Values

TRUE: if the port is open

FALSE: if the port is close.

1.7.6 SetCallback

In UHF RFID reader application program, to register callback delegate, that will be call at the end of operation to tag

```
RFID_RESULT    SetCallback (
        RfidCallbackProcCallbackProc
);
```

Parameters

CallbackProc

Callback delegate will get reply from module.

Return Values

RFID_RESULT_SUCCESS will be returned if all the processes are performed successfully,

Notes

RFID_RESULT_NOT_OPENED will be returned if to call function under not opened RFID reader module.

RFID_RESULT_INVALID_ARGS will be returned if CallbackProc is NULL.

1.7.7 GetResult

When the operation of reading data of UHF RFID tag has been done, the registered Callback delegate will be executed, and in the internal Callback delegate, application program will use this function to read Data or Reply.

```
RFID_RESULT    GetResult (
        String Result,
        RFID_CALLBACK_TYPE CallbackType,
        IntPtr wParam,
        IntPtr lParam
);
```

Parameters

Result

String parameter that will store Data or Reply that was read from UHF RFID tag.

CallbackType

set RFIDCALLBACKDATA CallbackType as it is that handed over while Callback delegate executed and application will accord the contents of CallbackType to process the Data or Reply.

wParam

Set RFIDCALLBACKDATA's wParam as it is which handed over while Callbackdelegate was executed.

lParam

Set RFIDCALLBACKDATA's IParam as it is which handed over while Callback delegate was executed.

Return Values

RFID_RESULT_SUCCESS will be returned if all the processes are performed successfully,

Notes

This function must be called within the delegate function that registered by user

1.7.8 ReadEpc

Read EPC data from UHF RFID tag.

```
RFID_RESULT   ReadEpc(
        bool bSyncMode,
        RFID_READ_TYPE ReadType,
        String EpcData
);
RFID_RESULTReadEpc(
        bool bSyncMode,
        RFID_READ_TYPE ReadType,
        ref RFIDMASKPARAMS Mask,
        String EpcData
);
```

Parameters

bSyncMode

Mode of operation of the function. (True=Sync, False=Async)

ReadType

Mode of Singe Tag Reading or Multiple Tag Reading

Mask

masking pattern which will be used in inventory. (masking pattern of nibble unit.)

EpcData

Variable that will store EPC value, when it is called as Sync Mode.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully

1.7.9 ReadEpcEx

Reading EPC data from Tag(Inventory)

Supporting MASK function of bit unit and SELECT function.

It has as same as the ReadEpc function, but could set Making pattern at bit unit or could

set select command

```
RFID_RESULT   ReadEpcEx(
        bool bSyncMode,
        RFID_READ_TYPE ReadType,
        String EpcData
);
RFID_RESULT   ReadEpcEx(
        bool bSyncMode,
        RFID_READ_TYPE ReadType,
        ref RFIDMASKPARAMS_EXMask,
        String EpcData
);
RFID_RESULT   ReadEpcEx(
        bool bSyncMode,
        RFID_READ_TYPE ReadType,
        ref RFIDSELMASKPARAMS_EX[] Masks,
        uintNumberOfMasks
        String EpcData
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async).

ReadType

Mode of Singe Tag Reading or Multiple Tag Reading

Mask

masking pattern of bit unit that will be used to inventory

Masks

array of making pattern of bit unit that will be used to inventory.

NumberOfMasks

The number of data of Masks

EpcData

Variable that will store EPC value, when it is called as Sync Mode

Return Values

RFID_RESULT_SUCCESS will be returned if performed.

Notes

If using the array of Masking pattern,the Sync mode is not supported.

Third overloaded function is supported only iftype of module is MODULE_TYPE_H1000.

1.7.10 Stop

Stop the operation which performing in UHF RFID reader module of Async mode

```
RFID_RESULT    Stop();
```

Parameters

Block

Mode of operation

True : waiting and returning until stop operation finishes.(up to 5 seconds)

False : transmitting stop command and function is returned immediately.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Second overloaded function is supported only if module type is MODUL_TYPE_H1000.

1.7.11 ReadMemBank

Read data at word (2Bytes) from UHF RFIED tag's specific memory bank.

```
RFID_RESULT    ReadMemBank(
        Bool bSyncMode,
        MEM_BANK MemBank,
        Uint nWordPtr,
        Uint nWordCount,
        Bool bContinuous,
        String AccessPassword,
        String MemBankData
);
RFID_RESULT    ReadMemBank(
        Bool bSyncMode,
        MEM_BANK MemBank,
        Uint nWordPtr,
        Uint nWordCount,
        Bool bContinuous,
        String AccessPassword,
        RefRFIDMASKPARAMS Mask,
        String MemBankData
);
```

Parameters

bSyncMode

Mode of operation of the function(True=Sync, False=Async)

MemBank

the type of memory bank which will be reading.

nWordPtr

starting offset within bank: Min. Value: 0 ~ Max. Value: 422820625

nWordCount

Data value which will be read and at word unit counting. Min Value: 1 ~ Max. Value: 255.

bContinuous

Continuous mode (True=Enable, False=Disable)

AccessPassword

access password that will be used when memory bank was locked.

Mask

Masking pattern of nibble unit that will be used to ReadMemBank.

MemBankData

variablewhereMembank value will be stored in sync mode.

Return Values

RFID_RESULT_SUCCESS will be returned if executed successfully.

1.7.12 ReadMemBankEx

Reading data in word(2bytes) unit from specific memory bank of Tag.

Supporting Mask function of bit unit and SELECT function.

```
RFID_RESULT   ReadMemBankEx(
        Bool bSyncMode,
        MEM_BANK MemBank,
        Uint nWordPtr,
        Uint nWordCount,
        Bool bContinuous,
        String AccessPassword,
        String MemBankData
);
RFID_RESULT   ReadMemBankEx(
        Bool bSyncMode,
        MEM_BANK MemBank,
        Uint nWordPtr,
        Uint nWordCount,
        Bool bContinuous,
```

```
        String AccessPassword,
        refRFIDMASKPARAMS_EX Mask,
        String MemBankData
);
RFID_RESULT   ReadMemBankEx(
        Bool bSyncMode,
        MEM_BANK MemBank,
        Uint nWordPtr,
        Uint nWordCount,
        Bool bContinuous,
        String AccessPassword,
        RFIDSELMASKPARAMS_EX[] Masks,
        Uint NumberOfMasks,
        String MemBankData
);
```

Parameters

bSyncMode

Mode of operation of the function(True=Sync, False=Async)

MemBank

the type of memory bank which will be reading.

nWordPtr

starting offset within bank: Min. Value: 0 ~ Max. Value: 422820625

nWordCount

Data value which will be read and at word unit counting. Min Value: 1 ~ Max. Value: 255.

bContinuous

Continuous mode (True=Enable, False=Disable)

AccessPassword

access password that will be used when memory bank was locked.

Mask

Masking pattern of bit unit

Masks

Alignment of masking pattern of bit unit.

NumberOfMasks

The number of data of Masks.

MemBankData

Variable where MemBank value will be stored in Sync Mode.

Return Values

RFID_RESULT_SUCCESS will be returned if executed successfully.

Notes

Continuous mode not supported in Sync mode.

Third overloaded function is supported only if module type is MODUL_TYPE_H1000

1.7.13 WriteMemBank

Recording data in word(2bytes) unit to memory bank of Tag

```
RFID_RESULT   WriteMemBank(
        Bool bSyncMode,
        MEM_BANK MemBank,
        Uint nWordPtr,
        String Data,
        Bool bContinuous,
        String AccessPassword,
);
RFID_RESULT WriteMemBank (
        Bool bSyncMode,
        MEM_BANK MemBank,
        Uint nWordPtr,
        String Data,
        Bool bContinuous,
        String AccessPassword,
        RefRFIDMASKPARAMS Mask,
);
```

Parameters

bSyncMode

Mode of operation of the function(True=Sync, False=Async)

MemBank

the type of memory bank which will be reading.

nWordPtr

starting offset within bank: Min. Value: 0 ~ Max. Value: 422820625

Data

the data which will be written into the memory bank.

bContinuous

Continuous mode (True=Enable, False=Disable)

AccessPassword

access password that will be used when memory bank was locked.

Mask

Masking pattern of bit unit

Return Values

RFID_RESULT_SUCCESS will be returned if command executed successfully.

Notes

Continuous mode not supported in Sync mode.

1.7.14 WriteMemBankEx

Recording data in word(2bytes) unit to specific memory Bank of Tag)

Supporting MASK function of bit unit and SELECT function.

```
RFID_RESULT   WriteMemBankEx (
        Bool bSyncMode,
        MEM_BANK MemBank,
        Uint nWordPtr,
        String Data,
        Bool bContinuous,
        String AccessPassword,
);
RFID_RESULT WriteMemBankEx (
        Bool bSyncMode,
        MEM_BANK MemBank,
        Uint nWordPtr,
        String Data,
        Bool bContinuous,
        String AccessPassword,
        refRFIDMASKPARAMS_EX Mask,
);
RFID_RESULT WriteMemBankEx (
        Bool bSyncMode,
        MEM_BANK MemBank,
        Uint nWordPtr,
        String Data,
        Bool bContinuous,
        String AccessPassword,
        RFIDSELMASKPARAMS_EX[] Masks,
        Uint NumberOfMasks
);
```

Parameters

bSyncMode

Mode of operation of the function(True=Sync, False=Async)

MemBank

the type of memory bank which will be reading.

nWordPtr

starting offset within bank: Min. Value: 0 ~ Max. Value: 422820625

Data

the data which will be written into the memory bank.

bContinuous

Continuous mode (True=Enable, False=Disable)

AccessPassword

access password that will be used when memory bank was locked.

Mask

Masking pattern of bit unit

Masks

Alignment of masking patter of bit unit.

NumberOfMasks

The number of data of masks.

Return Values

RFID_RESULT_SUCCESS will be returned if command executed successfully.

Notes

Continuous mode not supported in Sync mode.

Third overloaded function is supported only if module type is MODUL_TYPE_H1000

1.7.15 LockField

Lock the specific field of Tag. Access password is needed in access to locked field.

```
RFID_RESULT   LockFiled(
        Bool bSyncMode,
        refLOCKUNLOCKFIELD LockField,
        Bool bContinuous,
        String AccessPassword,
);
RFID_RESULT LockFiled (
        Bool bSyncMode,
        refLOCKUNLOCKFIELD LockField,
        Bool bContinuous,
        String AccessPassword,
        refRFIDMASKPARAMS Mask,
```

);

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

LockField

the field where will be locked.

bContinuous

Continuous mode. (True=Enable, False=Disable)

AccessPassword

access password

Mask

Masking pattern of nibble unit.

Return Values

RFID_RESULT_SUCCESS will be returned if command executed successfully.

Notes

Continuous mode not supported in Sync mode.

Third overloaded function is supported only if module type is MODUL_TYPE_H1000

1.7.16 LockFieldEx

Lock the specific field of Tag. Access password is needed in access to locked field.

Support MASK function of bit unit and SELECT function.

```
RFID_RESULT   LockFiledEx (
        Bool bSyncMode,
        refLOCKUNLOCKFIELD LockField,
        Bool bContinuous,
        String AccessPassword,
);
RFID_RESULT LockFiledEx (
        Bool bSyncMode,
        refLOCKUNLOCKFIELD LockField,
        Bool bContinuous,
        String AccessPassword,
        refRFIDMASKPARAMS_EX Mask,
);
RFID_RESULT LockFiledEx (
        Bool bSyncMode,
        refLOCKUNLOCKFIELD LockField,
```

```
    Bool bContinuous,
    String AccessPassword,
    refRFIDSELMASKPARAMS_EX[] Masks,
    uintNumberOfMasks
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

LockField

the field where will be locked.

bContinuous

Continuous mode. (True=Enable, False=Disable)

AccessPassword

access password

Mask

Masking pattern of nibble unit.

Masks

Alignment of masking pattern of bit unit.

NumberOfMasks

The number of data of Masks

Return Values

RFID_RESULT_SUCCESS will be returned if executed successfully.

Notes

Continuous mode not supported in Sync mode.

Third overloaded function is supported only if module type is MODUL_TYPE_H1000

1.7.17 UnLockField

Unlock the specific field of Tag

```
RFID_RESULT   UnLockFiled(
    Bool bSyncMode,
    refLOCKUNLOCKFIELD UnLockField,
    Bool bContinuous,
    String AccessPassword,
);
RFID_RESULT UnLockFiled (
    Bool bSyncMode,
    refLOCKUNLOCKFIELD UnLockField,
    Bool bContinuous,
```

```
    String AccessPassword,
    refRFIDMASKPARAMS Mask,
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

UnLockField

the field where will be unlocked.

bContinuous

Continuous mode. (True=Enable, False=Disable)

AccessPassword

access password

Mask

Masking pattern of nibble unit.

Return Values

RFID_RESULT_SUCCESS will be returned if command executed successfully.

Notes

Continuous mode not supported in Sync mode.

Third overloaded function is supported only if module type is MODUL_TYPE_H1000

1.7.18 UnLockFieldEx

Unlock the specific field of Tag

Support MASK function of bit unit and SELECT function.

```
RFID_RESULT   UnLockFiledEx (
    Bool bSyncMode,
    refLOCKUNLOCKFIELD UnLockField,
    Bool bContinuous,
    String AccessPassword,
);
RFID_RESULT UnLockFiledEx (
    Bool bSyncMode,
    refLOCKUNLOCKFIELD UnLockField,
    Bool bContinuous,
    String AccessPassword,
    refRFIDMASKPARAMS_EX Mask,
);
```

```
RFID_RESULT UnLockFiledEx (
    Bool bSyncMode,
    refLOCKUNLOCKFIELD UnLockField,
    Bool bContinuous,
    String AccessPassword,
    refRFIDSELMASKPARAMS_EX[] Masks,
    uintNumberOfMasks
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

UnLockField

the field where will be unlocked.

bContinuous

Continuous mode. (True=Enable, False=Disable)

AccessPassword

access password

Mask

Masking pattern of nibble unit.

Masks

Alignment of masking pattern of bit unit.

NumberOfMasks

The number of data of Masks.

Return Values

The same as UnLockField.

Return Values

RFID_RESULT_SUCCESS will be returned if command executed successfully.

Notes

Continuous mode not supported in Sync mode.

Third overloaded function is supported only if module type is MODUL_TYPE_H1000

1.7.19 Permalock

Cannot change the field's Secured state permanently which by lock the specific filed of PERMALOCK_FIELD.

Once locked the field, then cannot unlock the secured status,

Secured status means that the tag's password is active and there's need to input password while try to access tag.

```
RFID_RESULT    Permalock (
        Bool bSyncMode,
        PERMALOCK_FIELD PermalockField,
        Bool bSecured,
        Bool bContinuous,
        String AccessPassword,
);
RFID_RESULT Permalock (
        Bool bSyncMode,
        PERMALOCK_FIELD PermalockField,
        Bool bSecured,
        Bool bContinuous,
        String AccessPassword,
        refRFIDMASKPARAMS Mask,
);
```

Parameters

bSyncMode

mode of operation of the function.. (True=Sync, False=Async)

PermalockField

field which will apply to Permanent lock.

bSecured

if set as true, need to access with access password permanently.

If set as False, can access without access password permanently.

bContinuous

Continuous mode. (True=Enable, False=Disable)

AccessPassword

access password

Mask

masking patter of nibble unit.

Return Values

RFID_RESULT_SUCCESS will be returned if the command executed successfully.

Notes

Continuous mode not supported in Sync mode.

Attention to that once successfully performed this function; the state of Secured will notbe changed permanently.

1.7.20 PermalockEx

Permalock the specific field of Tag.

Support SELECT function and MASK function of bit unit.

```
RFID_RESULT   PermalockEx (
        Bool bSyncMode,
        refPERMALOCK_FIELD PermalockField,
        Bool bSecured,
        Bool bContinuous,
        String AccessPassword,
);
RFID_RESULT PermalockEx (
        Bool bSyncMode,
        refPERMALOCK_FIELDPermalockField,
        Bool bContinuous,
        String AccessPassword,
        refRFIDMASKPARAMS_EX Mask,
);
RFID_RESULT PermalockEx (
        Bool bSyncMode,
        refPERMALOCK_FIELDPermalockField,
        Bool bContinuous,
        String AccessPassword,
        RFIDSELMASKPARAMS_EX[] Masks,
        uintNumberOfMasks
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

PermalockField

Field that permalock will be applied to

bSecured

If set as True, there is need to access Tag with access password permanently.

bContinuous

Continuous mode. (True=Enable, False=Disable)

AccessPassword

access password

Mask

Masking pattern of bit unit.

Masks

Alignment of masking pattern of bit unit.

NumberOfMasks

The number of data of Masks..

Notes

Continuous mode not supported in Sync mode.

Third overloaded function is supported only if module type is MODUL_TYPE_H1000

1.7.21 KillTag

To kill the UHF RFID tag which in order to not response to the command of reader.

If the kill's password is "00000000", then cannot perform kill. In order to perform kill command, tag must be set with other password, not "00000000". Once tag be killed, it never recoveragain.

```
RFID_RESULT   KillTag (
        Bool bSyncMode,
        String KillPassword,
        Bool bContinuous,
);
RFID_RESULT KillTag (
        Bool bSyncMode,
        String KillPassword,
        Bool bContinuous,
        refRFIDMASKPARAMS Mask,
);
```

Parameters

bSyncMode

mode of operation of the function.(True=Sync, False=Async)

KillPassword

kill password saved in Tag

bContinuous

Continuous mode. (True=Enable, False=Disable)

Mask

masking pattern of nibble unit.

Return Values

RFID_RESULT_SUCCESS will be returned if command executed successfully,

Notes

Continuous mode is not supported in Sync mode.

Once performed this function that will never be get restored again.

1.7.22 KillTagEx

To kill the UHF RFID tag which in order to not response to the command of reader.

If the kill's password is "00000000", then cannot perform kill. In order to perform kill command, tag must be set with other password, not "00000000". Once tag be killed, it never recover again.

Support SELECT function and MASK function of bit unit.

```
RFID_RESULT   KillTagEx (
        Bool bSyncMode,
        String KillPassword,
        Bool bContinuous,
);
RFID_RESULT KillTagEx (
        Bool bSyncMode,
        String KillPassword,
        Bool bContinuous,
        refRFIDMASKPARAMS_EX Mask,
);
RFID_RESULT KillTagEx (
        Bool bSyncMode,
        String KillPassword,
        Bool bContinuous,
        RFIDSELMASKPARAMS_EX[] Masks,
        uintNumberOfMasks
);
```

Parameters

bSyncMode

mode of operation of the function.(True=Sync, False=Async)

KillPassword

kill password saved in Tag

bContinuous

Continuous mode. (True=Enable, False=Disable)

Mask

masking pattern of nibble unit.

Masks

Alignment of masking pattern of bit unit.

NumberOfMasks

The number of data of Masks..

Return Values

RFID_RESULT_SUCCESS will be returned if command executed successfully,

Notes

Continuous mode is not supported in Sync mode.

Once performed this function that will never be get restored again.

Third overloaded function is supported only if module type is MODUL_TYPE_H1000

1.7.23 SetDefault

Initialize the UHF RFID reader module values.

```
RFID_RESULT    SetDefault ();
```

Parameters

None

Return Values

RFID_RESULT_SUCCESS will be returned if perform properly.

1.7.24 EnableExtendedInformation

Enable or disable to read RSSI value additionally while reading Tag.

```
RFID_RESULT    EnableExtendedInformation (
        Bool bEnable
);
```

Parameters

bEnable

TRUE: set Extended Information Mode.

FALSE: remove setting of Extended Information Mode.

Return Values

RFID_RESULT_SUCCESS will be returned if perform properly.

1.7.25 IsRunning

Return whether Operation of RFID Module is progressing.

```
bool    IsRunning ( );
```

Parameters

None

Return Values

If module operation is in progress, True will be returned, and if not, False will be returned.

It can be supported only if module type is MODULE_TYPE_H1000

1.7.26 GetR1000MacReg

Reading register value of R1000

```
boolGetR1000MacReg (
    ushortReg,
    ref uint Value
);
```

Parameters

Reg

Register address (see separate document)

Value

Variable which register value will be stored in

Return Values

True will be returned, if performed successfully.

Notes

It can be supported only if module type is MODULE_TYPE_H1000

1.7.27 SetR1000MacReg

Recordvalue in register of R1000

```
boolSetR1000MacReg (
    ushortReg,
    uintValue
);
```

Parameters

Reg

Register address (see separate document)

Value

Variable which register value will be stored in

Return Values

True will be returned, if performed successfully.

Notes

It can be supported only if module type is MODULE_TYPE_H1000

1.7.28 SetAirDura

Set the operating time of Tx Cycle module.

```
RFID_RESULT    SetAirDura (
        AIRDURAPARAMS Value
);
```

Parameters

Value

Structure variable that stores On time and Off time of Tx Cycle.

Return Values

RFID_RESULT_SUCCESS will be returned, if performed successfully.

Notes

MODIf MODULE_TYPE_I2000, Tx Cycle operation time default is On (200ms), Off (0ms).

1.7.29 GetEuroMode

read the operating way of the module, if module type is MODULE_TYPE_H1000 and Europe.

```
boolGetEuroMode (
        ref uintmode
);
```

Parameters

mode

1 : four channels hopping

2 : one channel of 4 channels is selected by the random and fixed.

Return Values

RFID_RESULT_SUCCESS will be returned, if performed successfully.

Notes

1.7.30 SetEuroMode

Set the operating way of the module, if module type is MODULE_TYPE_H1000 and Europe.

```
boolSetEuroMode (
        uint mode
);
```

Parameters

mode

1 : four channels hopping

2 : one channel of 4 channels is selected by the random and fixed.

Return Values

RFID_RESULT_SUCCESS will be returned, if performed successfully.

Notes

1.8 Properties

1.8.1 FirmwareVersion

Read the firmware version of the UHF Module.

string DeviceID{ get; }

Value

UHF module's firmware version.

1.8.2 HoppingMode

Read the Hopping Mode which has set in the module.

HOPPING_MODE HoppingMode{ get; }

Value

Hopping Mode

1.8.3 InventoryTarget

Set or read the target while performing the ReadEpc.

INVENTORIED_STATE InventoryTarget{ get; set:}

Value

Inventory Target

1.8.4 LBTChannelState

Compose or read LBT Channel by user (Apply to Japan, Europe type)
the frequency of use and the number of channels of Each Module is shown in the table below.

MODULE_TYPE_H1000		MODULE_TYPE_I2000	
Europe	Japan	Japan 1W	Japan 250mW
865.7	X	916.8	916.8
866.3	952.4	918.0	918.0
866.9	952.6	919.2	919.2
867.5	952.8	920.4	920.4

	953.0	920.6	920.6
	953.2	920.8	920.8
	953.4		921.0
	953.6		921.2
	953.8		921.4
	X		921.6
			921.8
			922.0
			922.2
			922.4
			922.6
			922.8
			923.0
			923.2
			923.4

uint LBTChannelState{ get; set:}

Value

LBT Channel State Min.1 ~ Max.:255

When change this value to binary numeral, the least significant bit will be Channel1 and if bit is 1, use that channel.

1.8.5 ChannelState

you can use ChannelState instead of LBTChannelState in order to channel setting, if module type is MODULE_TYPE_H1000 and Europe

uintChannelState{ get; set:}

Value

Channel State

1.8.6 LBTTime

Set or read the time, which to occupy a channel and to read Tag. (apply to Japan, Europe Type). And the units is ms and according to the radio regulations the occupy time cannot exceed 4 seconds.

uint LBTTime{ get; set:}

Value

LBT Time Min: 0 ~ Max: 4000

Notes

It can be supported if module type is MODULE_TYPE_H1000

1.8.7 OperationTime

Set or read the UHF module's command execution time. The unit as sec. and if it be 0, the command executed completely or stand by till get Stop command from user.

```
uint   OperationTime{ get; set:}
```

Value

Operaion Time Min: 0 ~ Max: 60, basic value: 0

1.8.8 PowerLevel

Set or read UHF Module's Power Level.

```
uint   PowerLevel{ get; set:}
```

Value

Power Level Min: 0 ~ Max: 31, basic value: 0(about 28dBm)

Notes

Record the power level value which will be reduced by full power (about 28dBm)

Ex) Powerlevel = 3, (28-3=25) then the will be set power level is 25dBm.

1.8.9 ProtocolVersion

Read the protocol version of the UHF Module.

```
string   ProtocolVersion{ get; }
```

Value

Protocol Version

1.8.10 QValue

Set or read the Q Parameter value that is used at ReadEpc.

uint QValue{ get; set;}

Value

QValue Min: 0 ~ Max: 15, basic value: 5

1.8.11 Session

Set or read the Session value that will be used at ReadEpc.

SESSION_TYPE Session{ get; set;}

Value

Session Type

1.8.12 Selects

Select Command number of execution.

intSelects{ get; set;}

Value

Number of execution.

Notes

even if you set with xxxEx() function, if this value is 0,SELECT command will not be executed. and Even if you do not set the SELECT command, if this value is more than 1, SELECT command will be run about all Tag.

It can be supported if module type is MODULE_TYPE_H1000

1.8.13 AirDuty

Set Tx cycle of RFID device.

intAirDuty{ set;}

Value

Speed value. (Min:10 ~ Max:100)

As the value increases, the speed increases.

Notes

Don't use this function for general purpose.The optimal value is already set about battery consumption, heat generation, etc.

It can be supported only if module type is MODULE_TYPE_H1000

1.8.14 ModuleType

Return type of RFID module equipped with PDA.

```
uint   ModuleType{ get; }
```

Value

Type RFID module.

Notes

There are differences in some functions, according to type of RFID Module.

Reference to 1.2 Constants

1.8.15 MaxPwr

Return maximum value of Tx power of RFID module equipped with PDA.

```
uint   MaxPwr{ get; }
```

Value

30x : 1W

28x : 600mW

27x : 500mW

24x : 250mW

1.8.16 EnableTagFocus

Function operated only in MODULE_TYPE_I2000.

Enable/Disable Impinj Extension(Tag Focus) function.

InventoryTarget, Session properties cannot be changed, while EnableTagFocus property is set to true.

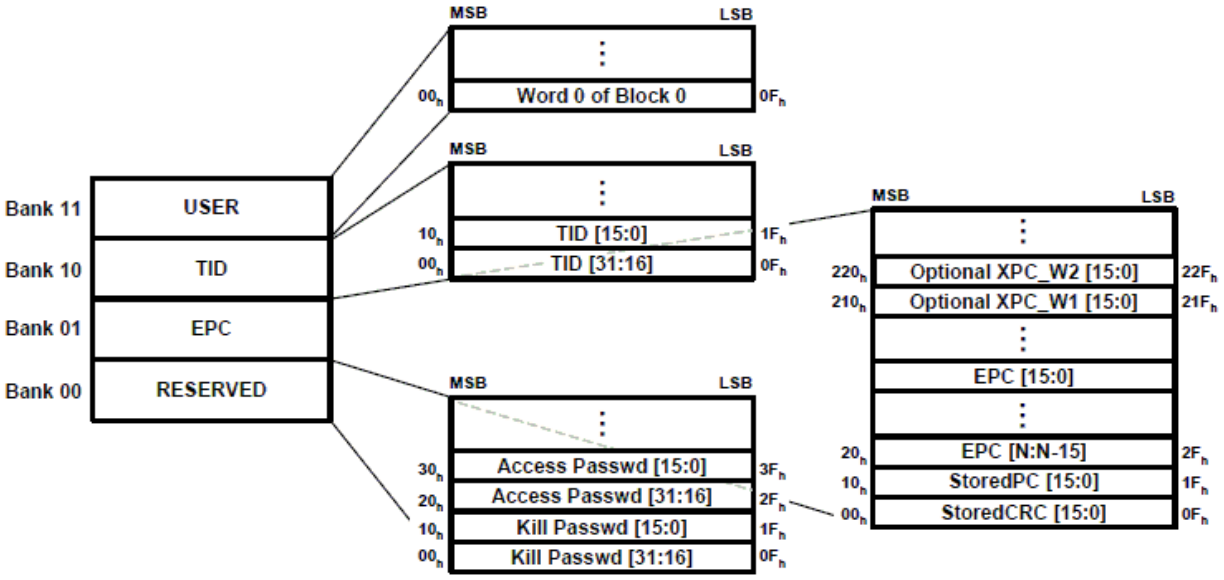
```
bool   EnableTagFocus{ get; set;}
```

Value

Enable/Disable state.

2 C/C++ API Reference

- ISO 18000 6C(EPC Class 1 Generation 2) Logical Memory Map



2.1 Enumerations

2.1.1 RFID_RESULT

The result of a call to functions.

- **RFID_RESULT_SUCCESS**
  Function executed successfully.
- **RFID_RESULT_OUTOFMEMORY**
  Failed to assign resource.
- **RFID_RESULT_INVALID_ARGS**
  Invalid parameter.
- **RFID_RESULT_FAILURE**
  Function execution failed.
- **RFID_RESULT_UNEXPECTED**
  Undefined Error.
- **RFID_RESULT_ALREADY_OPENED**
  RFID device port has already opened.
- **RFID_RESULT_INVALID_DEVICE**
  Do not support UHF reader type currently.
- **RFID_RESULT_NOT_CONNECTED**
  Module doesn't work properly, after be opened.
- **RFID_RESULT_NOT_DETECT**
  Cannot find Tag and command aborted.
- **RFID_RESULT_ACCESS_ERROR**
  Try to access with incorrect password, or try to access nonexistent region.
- **RFID_RESULT_NOT_OPENED**
  Call function without open.
- **RFID_RESULT_COMMAND_ERROR**
  Happened error while executing command, or received another command before finished one command.
- **RFID_RESULT_LOW_BATTERY**
  Module cannot work, because of low battery.
- **RFID_RESULT_UNKNOWN**
  UHF reader module happen Unknown Error.
- **RFID_RESULT_NOT_SUPPORTED**
  Not supported command currently.

- **RFID_RESULT_STOPPED**
  UHF reader module command aborted.
- **RFID_RESULT_POWER_OFF**
  UHF reader module power turned off.

2.1.2 RFIDMEM_BANK

Memory Bank of Tag which RFID device will access

- **RFID_MEM_BANK_RESERVED**
  memory where will be stored tag kill or access password.
- **RFID_MEM_BANK_EPC**
  memory where will be stored StoredCRC, EPC, XPC information.
- **RFID_MEM_BANK_TID**
  ISOmemory where will be stored ISO/IEC 15963 allocation class identifier, Tag serial number and so on.
- **RFID_MEM_BANK_USER**
  user specific data storage.

2.1.3 RFIDREAD_TYPE

Single, multiple mode that will be applied when reading Tag(Inventory)

- **RFID_EPC_GEN2_MULTI_TAG**
  Multiple read
- **RFID_EPC_GEN2_ONE_TAG**
  Single read.

2.1.4 RFIDPERMALOCK_FIELD

Field where Permalock will be applied.

- **RFID_PERMALOCK_FIELD_ACCESS_PASSWORD**
  Tag's Access password field
- **RFID_PERMALOCK_FIELD_KILL_PASSWORD**
  Tag's Kill password field
- **RFID_PERMALOCK_FIELD_EPC**
  Tag's EPC bank
- **RFID_PERMALOCK_FIELD_TID**
  Tag's TID bank
- **RFID_PERMALOCK_FIELD_USER**

Tag's USER bank

2.1.5 RFIDCALLBACKTYPE

A value indicating the reason why Callback delegate is running

- **RFIDCALLBACKTYPE_DATA**
  because of module has already read memory data from tag, so it means that the Callback delegate has been executed.
- **RFIDCALLBACKTYPE_REPLY**
  because received the results of command execution from module, so it means that Callback delegate has been executed.

2.1.6 RFIDINVENTORIEDSTATE

Inventoried state of tag which will be used as target while UHF module executing ReadEpc(Inventory).

- **RFID_INVENTORIEDSTATE_A**
  Inventoried state A
- **RFID_INVENTORIEDSTATE_B**
  Inventoried state B
- **RFID_INVENTORIEDSTATE_AB**
  Inventoried state A or B

2.1.7 RFIDSESSIONTYPE

Session which will be used while UHF module perform ReadEpc.

- **RFID_SESSIONTYPE_0**
  S0
- **RFID_SESSIONTYPE _1**
  S1
- **RFID_SESSIONTYPE _2**
  S2
- **RFID_SESSIONTYPE _3**
  S3

2.2 Constants

2.2.1 WM_RFID_RESPONSE

Value that will inform the result of RFID when WM_COPYDATA message occurs. If the value of COPYDATASTRUCT dwData is same with WM_RFID_RESPONSE, the results about RFID has arrived.

```
#define WM_RFID_RESPONSE        WM_USER + 1804
```

2.2.2 MAX_MASKS_BYTES

Maximum byte that can be used as Masking pattern.

- **#defineMAX_MASKS_BYTES 32**

2.2.3 MODULE_TYPE_NONE

RFID module is not equipped.

- **#defineMODULE_TYPE_H1000 0x30**

2.2.4 MODULE_TYPE_H1000

January 2013 now, RFID module that corresponds to the country (excluding Japan)

#defineMODULE_TYPE_H1000 0x31

2.2.5 MODULE_TYPE_I2000

January 2013 now, Japan only, RFID module that corresponds to Japan.
the changed Radio Law of JapanApplied

#defineMODULE_TYPE_I2000 0x32

2.2.6 MODULE_TYPE_I900

January 2013 now, RFID Module that can be mounted on only AT980

#defineMODULE_TYPE_I900 0x33

2.3 Reply Words List

If executing function of Tag Operation without using SyncMode, the following string.will be shown.

- **"Other Error"**
  Other error.

- **"Memory Overrun"**
  Tryaccessinginexistent memory region.

- **"Memory Locked"**
  Tag is locked.

- **"Insufficient Tag Power"**
  Cannot execute Write command because the tag power is insufficient.

- **"Non-specific Error"**
  unknown error occurs.

- **"Check Antenna"**
  Cannot connect Antenna with RFID device.

- **"Try after cooled"**
  Module is overheating.

- **"Insufficient PDA Power"**
  Failed to execute Function because of the lack of main battery

- **"Not Supported"**
  Unsupported function currently.

- **"Not Connected"**
  Module cannot work properly, after be opened.

- **"Not Opened"**
  Call function without opening function..

- **"Bad Access Password"**
  Try to access tag with incorrect Access Password.

- **"Invalid Parameter"**
  invalid parameter

- **"Command Error"**
  Happened error while executing command, or received another command before finished one command.

- **"Success"**
  Function executed successfully.

- **"Not Detect"**
  Command finished before detect Tag.

- **"Multi Read Stop"**
  ReadEpc(multiple) command is stopped by the stop command.

- **"EAS"**
  Detect EAS of NXP Tag.

2.4 Structures

2.4.1 RFIDLOCK_UNLOCK_FIELD

Store information that about lock or unlock each field of tag.
Lock or unlock only apply to set with True's member.

```
typedef struct
{
   bool bAccessPassword;
   bool bEPC;
   bool bKillPassword;
   bool bTID;
   bool bUSER;
}RFIDLOCK_UNLOCK_FIELD, *RFIDPLOCK_UNLOCK_FIELD;
```

- **bAccessPassword**
  Tag's access password field
- **bEPC**
  Tag's EPC bank
- **bKillPassword**
  Tag's kill password field
- **bTID**
  Tag's TID bank
- **bUSER**
  Tag's USER bank.

Ex) in order to Access Password and Lock or Unlock EPC, to structure as below:

```
LOCK_UNLOCKFIELD Lock;
Lock.bAccessPassword = true;
Lock.bEPC = true;
Lock.bKillPassword = false;
Lock.bTID = false;
Lock.bUSER = false;
```

2.4.2 RFIDMASK_PARAMS

While call the command which apply to the tag, only make those satisfied relevant condition tags respond.

```
typedef struct
{
   BYTE MemBank;
   UINT nOffSet;
   LPWSTRszMaskPattern;
}RFIDMASK_PARAMS, *RFIDPMASK_PARAMS;
```

- **MemBank**
  Memory Bank of tag which will be used as Mask.
- **nOffSet**
  Nibble offset of selected memory bank.
- **szMaskPattern**
  Mask pattern string which designated at 4bit.

Ex) when a tag's TID bank is "E2006004015F325A", and would like to use this TID bank as mask, the structure as below:

```
MASK_PARAMS Mask;
Mask.MemBank = MEM_BANK.TID;
Mask.nOffSet = 1;
Mask.MaskPattern = "2006";
```

또는

```
MASK_PARAMS ;
Mask.MemBank = MEM_BANK.TID;
Mask.nOffSet = 11;
Mask.MaskPattern = "F325";
```

2.4.3 RFIDMASK_PARAMS_EX

Using the function,while bit unit is masking pattern.

```
typedef struct
{
   BYTE MemBank;
   UINT nOffSet;
   UINT nBits;
   BYTEbyMaskPattern[MAX_MASK_BYTES];
}RFIDMASK_PARAMS_EX, *RFIDPMASK_PARAMS_EX;
```

- **MemBank**
  Tag's Memory Bank which will be used as Mask.
- **nOffSet**

Selected Memory Bank's bit offset.

- **nBits**

  bit of MaskPattern which will be used as masking pattern.

- **byMaskPattern**

  bit array which will be used as masking pattern.

2.4.4 RFID_MASK_PARAMS_EX2

Used when apply SELECT command using Masking pattern of Bit unit.

```
typedef struct
{
  BYTE ActionCode;
  UINTmskBits;
  BYTEmskPattern[MAX_MASK_BYTES];
  BYTEmskMemBank;
  UINTmskOffSet;
  BYTE SelectTarget;
}RFID_MASK_PARAMS_EX2, *PRFID_MASK_PARAMS_EX2;
```

- **ActionCode**

  Content that will apply to Target Flag of Tag

- **mskBits**

  bit of MaskPattern which will be used as masking pattern.

- **mskPattern**

  bit array which will be used as masking pattern.

- **mskMemBank**

  Tag's Memory Bank which will be used as Mask.

- **mskOffSet**

  Selected Memory Bank's bit offset.

- **SelectTarget**

  Set Target Flag of Tag which will be applied to the Select command.

  You can change tag's state flag with below option.

	If SELECT_TARGET is4(SL)		If SELECT_TARGET are between 0 ~ 3	
	Matching	Non-matching	Matching	Non-matching
ActionCode is 0	Assert SL	De-assert SL	→ A	→ B
ActionCode is 1	Assert SL	No Action	→ A	No Action
ActionCode is 2	No Action	De-assert SL	No Action	→ B
ActionCode is 3	Negate SL	No Action	Negate	No Action
ActionCode is 4	De-assert SL	Assert SL	→ B	→ A
ActionCode is 5	De-assert SL	No Action	→ B	No Action

ActionCode is 6	No Action	Assert SL	No Action	→ A
ActionCode is 7	No Action	Negate SL	No Action	Negate

2.4.5 RFIDCALLBACKDATA

Structure that will be received through callback function when receiving response from RFID device.

If Callback function is called, data will be read using RfidGetResult function with members of this structure.

```
typedef struct
{
  RFIDCALLBACKTYPE CallbackType;
  LPARAMlParam
  WPARAM wParam;
}RFIDCALLBACKDATA;
```

- **CallbackType**
  the executing reason of Callback function(Data or Reply)
  it must be transmitted to parameter of RfidGetResult
- **lParam**
  Parameter 1 which in order to read value from RFID module.
  it must be transmitted to parameter of RfidGetResult
- **wParam**
  Parameter 2 which in order to read value from RFID module.
  it must be transmitted to parameter of RfidGetResult

2.4.6 RFIDAIRDURAPARAMS

Structure used to set the operating time of Tx cycle Module.

```
typedef struct
{
  UINT nOnMs;
  UINT nOffMs;
}RFIDAIRDURAPARAMS;
```

- **nOnMs**
  operating time in Tx Cycle(millisecond)
  Min:0ms ~ Max:4000ms
- **nOffMs**
  break time in Tx Cycle(millisecond)
  Min:0ms ~ Max:4000ms

2.5 Callback function definition

2.5.1 RFIDCALLBACK

Callback function, that will process data while get reply from module.

typedef void (CALLBACK* **RFIDCALLBACK)**(RFIDCALLBACKDATA*pCallbackData);

- **pCallbackData**

pointer of RFIDCALLBACKDATA, which including data of getting from module.

2.6 Methods

2.6.1 RfidPowerOn

Put voltage into the UHF RFID reader module.

```
RFID_RESULTRfidPowerOn();
```

Parameters

None

Return Values

RFID_RESULT_SUCCESS will be returned ifperformed successfully.

Notes

You should call it in order to use UHF RFID device.

2.6.2 RfidPowerOff

Remove the power from UHF RFID reader module.

```
voidRfidPowerOff();
```

Parameters

None

Return Values

None

Notes

If you finish using UHF RFID device, turn off the power calling this function.

2.6.3 RfidOpen.

Allocating system resources, Opening the port of UHF RFID device.

```
RFID_RESULTRfidOpen();
```

Parameters

None

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.4 RfidClose

Deallocating resources, and close the port of UHF RFID device.

```
void    Close();
```

Parameters

None

Return Values

None

2.6.5 RfidIsOpened

Checking whether the port of UHF RFID device was open.

```
BOOLRfidIsOpened()
```

Parameters

None

Return Values

TRUE: if the port is open

FALSE: if the port is close.

2.6.6 RfidSetCallback

Register the callback function for receiving reply/data from RFID device.

```
RFID_RESULTRfidSetCallback (
        RFIDCALLBACKpCallback
);
```

Parameters

pCallback

RFIDCALLBACK callback function

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

You should call it after calling RfidOpen()

2.6.7 RfidSetHwnd

When you want process the results receiving windows messages instead of Callback function, set the application windows handle.

```
RFID_RESULT    RfidSetHwnd (
        HWND hWnd
);
```

Parameters

hWnd

application windows handle

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

It must be called after RfidOpen() is called.

2.6.8 RfidGetResult

Reading response from Tag Data or RFID device.

```
RFID_RESULTRfidGetResult (
        LPWSTRszResult,
        RFIDCALLBACKTYPE CallbackType,
        WPARAMwParam,
        LPARAMlParam
);
```

Parameters

szResult

variable that will store Data or Reply that was read from UHF RFID tag.

CallbackType

set RFIDCALLBACKDATA CallbackType as it is that handed over while Callback delegate executed and application will check whether calling of callback function is data or reply confirming contents of CallbackType of RFIDCALLBACKDATA and process according to type.

wParam

Set RFIDCALLBACKDATA's wParam as it is which handed over while Callback function was called.

lParam

Set RFIDCALLBACKDATA's lParam as it is which handed over while Callback delegate

was called.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

This function must be called within the callback function that registered by user

2.6.9 RfidReadEpc

Read EPC data from UHF RFID tag.

```
RFID_RESULTRfidReadEpc(
    BOOLbSyncMode,
    RFIDREAD_TYPE ReadType,
    RFIDPMASK_PARAMS pMask,
    LPWSTRszEpcData
);
```

Parameters

bSyncMode

mode of operation of the function (True=Sync, False=Async)

ReadType

Mode of Singe Tag Reading or Multiple Tag Reading

pMask

masking pattern which will be used to Inventory (masking pattern of nibble unit.)

szEpcData

Variable where EPC value will be stored when calling it as Sync mode.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.10 RfidReadEpcEx

Read EPC data from UHF RFID tag.(Inventory)

Support MASK function of bit units.

```
RFID_RESULTRfidReadEpcEx(
    BOOLbSyncMode,
    RFIDREAD_TYPE ReadType,
    RFIDPMASK_PARAMS_EX pMask,
    LPWSTRszEpcData
);
```

bSyncMode

mode of operation of the function (True=Sync, False=Async)

ReadType

Mode of Singe Tag Reading or Multiple Tag Reading

pMask

masking pattern which will be used to Inventory (masking pattern of nibble unit.)

EpcData

Variable where EPC value will be stored when calling it as Sync mode.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.11 RfidReadEpcEx2

Read EPC data from UHF RFID tag.(Inventory)

Support MASK function of bit units.

```
RFID_RESULT    RfidReadEpcEx2(
    BOOL bSyncMode,
    RFIDREAD_TYPE ReadType,
    PRFID_MASK_PARAMS_EX2 pMasks,
    UINT nNumberOfMasks,
    LPWSTR szEpcData
);
```

Parameters

bSyncMode

mode of operation of the function (True=Sync, False=Async)

ReadType

Mode of Singe Tag Reading or Multiple Tag Reading

pMask

address of masking pattern of bit units.

nNumberOfMasks

The number of data of pMasks.

EpcData

Variable where EPC value will be stored when calling it as Sync mode.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

Sync mode is not supported in using pMasks.

It can be supported only if module type is MODULE_TYPE_H1000

2.6.12 RfidStop

Stop the operation which performing in UHF RFID reader module of Asyn mode.

```
RFID_RESULTRfidStop();
```

Parameters

None

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.13 RfidStopEx

Stop the operation which performing in Asyn mode of UHF RFID device.

```
RFID_RESULT    RfidStop(
        BOOL block
);
```

Parameters

Block

Mode of operaton

TRUE : wait until stop operation finishes and reture(up to 5 seconds)

FALSE : transmit stop command and the function is returned immediately.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

It can be supported only if module type is MODULE_TYPE_H1000

2.6.14 RfidReadMemBank

Read data at word (2Bytes) from UHF RFIED tag's specific memory bank.

```
RFID_RESULTRfidReadMemBank(
        BOOLbSyncMode,
        RFIDMEM_BANK MemBank,
        UINTnWordPtr,
        UINTnWordCount,
        BOOLbContinuous,
        LPWSTRszAccessPassword,
        RFIDPMASK_PARAMS pMask,
        LPWSTRszMemBankData
```

);

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

MemBank

Memory Bank Type to be read

nWordPtr

starting offset to be read in Memory Bank (Min:0 ~ Max:422820625)

nWordCount

data size to be read. (Min:0 ~ Max:255)

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password which will be used when memory bank is locked

pMask

masking pattern of nibble units

szMemBankData

Variable where MemBank value will be stored when it is called as Sync mode.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

Continuous mode is not supported in Sync mode.

2.6.15 RfidReadMemBankEx

Read data at word (2Bytes) from UHF RFIED tag's specific memory bank.

Support Mask function of bit units

```
RFID_RESULTRfidReadMemBankEx(
    BOOLbSyncMode,
    RFIDMEM_BANK MemBank,
    UINT nWordPtr,
    UINTnWordCount,
    BOOLbContinuous,
    LPWSTRszAccessPassword,
    RFIDPMASK_PARMAS_EX pMask,
    LPWSTRszMemBankData
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

MemBank

Memory Bank Type to be read

nWordPtr

starting offset to be read in Memory Bank (Min:0 ~ Max:422820625)

nWordCount

data size to be read. (Min:0 ~ Max:255)

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password which will be used when memory bank is locked

pMask

masking pattern of nibble units

szMemBankData

Variable where MemBank value will be stored when it is called as Sync mode.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

Continuous mode is not supported in Sync mode.

2.6.16 RfidReadMemBankEx2

Read data at word (2Bytes) from UHF RFIED tag's specific memory bank.

Support SELECT function and Mask function of bit units.

```
RFID_RESULT    RfidReadMemBankEx2(
    BOOL bSyncMode,
    RFIDMEM_BANK MemBank,
    UINT nWordPtr,
    UINT nWordCount,
    BOOL bContinuous,
    LPWSTR szAccessPassword,
    PRFID_MASK_PARMAS_EX2 pMasks,
    UINT nNumberOfMasks,
    LPWSTR szMemBankData
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

MemBank

Memory Bank Type to be read

nWordPtr

starting offset to be read in Memory Bank (Min:0 ~ Max:422820625)

nWordCount

data size to be read. (Min:0 ~ Max:255)

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password which will be used when memory bank is locked

pMask

masking pattern of nibble units

nNumberOfMasks

The number of data of pMasks.

szMemBankData

Variable where MemBank value will be stored when it is called as Sync mode.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

If you use pMasks, Sync mode is not supported.

It can be supported only if module type is MODULE_TYPE_H1000

2.6.17 RfidWriteMemBank

Write the data into theUHF RFID tag specific memory bank at word (2Bytes)

```
RFID_RESULTRfidWriteMemBank (
    BOOLbSyncMode,
    RFIDMEM_BANK MemBank,
    UINTnWordPtr,
    LPWSTRszData,
    BOOLbContinuous,
    LPWSTRszAccessPassword,
    RFIDPMASK_PARAMSpMask,
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

MemBank

Memory Bank Type to be written

nWordPtr

starting offset to be written in Memory Bank (Min:0 ~ Max:422820625)

szData

data to be written.

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password which will be used when memory bank is locked

pMask

masking pattern of nibble units

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

Continuous mode is not supported in Sync mode.

2.6.18 RfidWriteMemBankEx

Write the data into theUHF RFID tag specific memory bank at word (2Bytes)

Support Mast function of bit units

```
RFID_RESULTRfidWriteMemBankEx (
    BOOLbSyncMode,
    RFIDMEM_BANK MemBank,
    UINTnWordPtr,
    LPWSTRszData,
    BOOLbContinuous,
    LPWSTRszAccessPassword,
    RFIDPMASK_PARAMS_EX pMask
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

MemBank

Memory Bank Type to be written

nWordPtr

starting offset to be written in Memory Bank (Min:0 ~ Max:422820625)

szData

data to be written.

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password which will be used when memory bank is locked

pMask

masking pattern of nibble units

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

Continuous mode is not supported in Sync mode.

2.6.19 RfidWriteMemBankEx2

Write the data into theUHF RFID tag specific memory bank at word (2Bytes)
Support SELECT function and MASK function of bit units.

```
RFID_RESULT    RfidWriteMemBankEx2 (
        BOOL bSyncMode,
        RFIDMEM_BANK MemBank,
        UINT nWordPtr,
        LPWSTR szData,
        BOOL bContinuous,
        LPWSTR szAccessPassword,
        PRFID_MASK_PARAMS_EX2pMasks,
        UINT nNumberOfMasks
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

MemBank

Memory Bank Type to be written

nWordPtr

starting offset to be written in Memory Bank (Min:0 ~ Max:422820625)

szData

data to be written.

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password which will be used when memory bank is locked

pMask

masking pattern of nibble units

nNumberOfMasks

The number of data of pMasks

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

If you use pMasks, Sync mode is not supported.

It can be supported if module type is MODULE_TYPE_H1000

2.6.20 RfidLockField

Lock the field which set as true among the member field of pLockField,
And there's need a access password when try to access the locked field.

```
RFID_RESULTRfidLockFiled (
    BOOLbSyncMode,
    RFIDPLOCK_UNLOCK_FIELDpLockField,
    BOOLbContinuous,
    LPWSTRszAccessPassword,
    RFIDPMASK_PARAMSpMask,
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

pLockField

field which lock will be applied to

bContinuous

Continuous mode. (True=Enable, False=Disable)

AccessPassword

access password

pMask

masking pattern of nibble units

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

Continuous mode is not supported in Sync mode.

2.6.21 RfidLockFieldEx

Lock the field which set as true among the member field of pLockField,
And there's need a access password when try to access the locked field.
Support Mask function of bit units.

```
RFID_RESULTRfidLockFiledEx (
    BOOLbSyncMode,
    RFIDPLOCK_UNLOCK_FIELDpLockField,
    BOOLbContinuous,
    LPWSTRszAccessPassword,
    RFIDPMASK_PARAMS_EXpMask,
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

pLockField

field which lock will be applied to

bContinuous

Continuous mode. (True=Enable, False=Disable)

AccessPassword

access password

pMask

masking pattern of nibble units

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

Continuous mode is not supported in Sync mode.

2.6.22 RfidLockFieldEx2

Lock the field which set as true among the member field of pLockField,
And there's need a access password when try to access the locked field.
Support SELECT function and MASK function of bit units.

```
RFID_RESULTRfidLockFiledEx2 (
    BOOL bSyncMode,
    RFIDPLOCK_UNLOCK_FIELD pLockField,
    BOOL bContinuous,
    LPWSTR szAccessPassword,
    PRFID_MASK_PARAMS_EX2 pMasks,
```

UINT nNumberOfMasks
);

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

pLockField

field which lock will be applied to

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password

pMask

address of masking pattern of bit units

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

If you use pMasks, Sync mode is not supported.

It can be supported if module type is MODULE_TYPE_H1000

2.6.23 RfidUnLockField

Unlock the specific field of Tag.

```
RFID_RESULTRfidUnLockFiled (
    BOOLbSyncMode,
    RFIDPLOCK_UNLOCK_FIELDpLockField,
    BOOLbContinuous,
    LPWSTRszAccessPassword,
    RFIDPMASK_PARAMSpMask,
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

pLockField

field which unlock will be applied to

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password

pMask

masking pattern of nibble units

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

Continuous mode is not supported in Sync mode.

2.6.24 RfidUnLockFieldEx

Unlock the specific field of Tag

Support Mast function of bit units.

```
RFID_RESULTRfidUnLockFiledEx (
    BOOLbSyncMode,
    RFIDPLOCK_UNLOCK_FIELDpLockField,
    BOOLbContinuous,
    LPWSTRszAccessPassword,
    RFIDPMASK_PARAMS_EXpMask,
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

pLockField

field which unlock will be applied to

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password

pMask

masking pattern of bit units

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

Continuous mode is not supported in Sync mode.

2.6.25 RfidUnLockFieldEx2

Unlock the specific field of Tag

Support SELECT function and MASK function of bit units.

```
RFID_RESULT RfidUnLockFiledEx2 (
```

```
    BOOL bSyncMode,
    RFIDPLOCK_UNLOCK_FIELD pLockField,
    BOOL bContinuous,
    LPWSTR szAccessPassword,
    PRFID_MASK_PARAMS_EX2 pMasks,
    UINT nNumberOfMasks
);
```

Parameters

bSyncMode

mode of operation of the function. (True=Sync, False=Async)

pLockField

field which unlock will be applied to

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password

pMask

address of masking pattern of bit units

nNumberOfMasks

The number of data of pMasks

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

If you use pMasks, Sync mode is not supported.

It can be supported if module type is MODULE_TYPE_H1000

2.6.26 RfidPermalock

Permalock the specific field of Tag.

Permanently locked field cannot remove secured status with unlock.

Secured status means that password is required to access tag because password of tag is enabled

```
RFID_RESULTRfidPermalock (
    BOOLbSyncMode,
    RFIDPERMALOCK_FIELD PermalockField,
    BOOLbSecured,
    BOOLbContinuous,
    LPWSTRszAccessPassword,
```

RFIDPMASK_PARAMSpMask,
);

Parameters

bSyncMode

mode of operation of function. (True=Sync, False=Async)

PermalockField

field which permalock will be applied to

bSecured

Scured status

If set as true, you have to access tag with access password.

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password.

pMask

masking pattern of nibble units.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully..

Notes

Continuous mode is not supported in Sync mode.

Attention to that once successfully performed this function; the state of Secured will be not changed permanently.

2.6.27 RfidPermalockEx

Permalock the specific field of Tag

RFID_RESULTRfidPermalockEx (
BOOLbSyncMode,
RFIDPERMALOCK_FIELD PermalockField,
BOOL bSecured,
BOOLbContinuous,
LPWSTRszAccessPassword,
RFIDPMASK_PARAMS_EXpMask,
);

Parameters

bSyncMode

mode of operation of function. (True=Sync, False=Async)

lockField

field which permalock will be applied to

bSecured

Scured status

If set as true, you have to access tag with access password.

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password.

pMask

masking pattern of bit units.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully..

Notes

Continuous mode is not supported in Sync mode.

2.6.28 RfidPermalockEx2

Permalock the specific field of Tag

Support SELECT function and Mask function of bit units

```
RFID_RESULT RfidPermalockEx2 (
    BOOL bSyncMode,
    RFIDPERMALOCK_FIELD PermalockField,
    BOOL bSecured,
    BOOL bContinuous,
    LPWSTR szAccessPassword,
    PRFID_MASK_PARAMS_EX2 pMassk,
    UINT nNumberOfMasks
);
```

Parameters

bSyncMode

mode of operation of function. (True=Sync, False=Async)

lockField

field which permalock will be applied to

bSecured

Scured status

If set as true, you have to access tag with access password.

bContinuous

Continuous mode. (True=Enable, False=Disable)

szAccessPassword

access password.

pMask

address of masking pattern of bit units.

nNumberOfMasks

The number of data of pMasks.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully..

Notes

If you use pMasks, Sync mode is not supported.

It can be supported if module type is MODULE_TYPE_H1000

2.6.29 RfidKillTag

killing the UHF RFID tag which in order to not response to the command of reader.

If the kill's password is "00000000", then cannot perform kill. In order to perform kill command, tag must be set with other password, not "00000000". Once tag be killed, it never recover again.

```
RFID_RESULTRfidKillTag (
    BOOLbSyncMode,
    LPWSTRszKillPassword,
    BOOLbContinuous,
    RFIDPMASK_PARAMSpMask
);
```

Parameters

bSyncMode

mode of operation of function. (True=Sync, False=Async)

szKillPassword

Kill Password saved in Tag

bContinuous

Continuous mode. (True=Enable, False=Disable)

Mask

masking pattern of nibble units..

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully..

Notes

Continuous mode is not supported in Sync mode.

Once performed this function that will never be get restored again.

2.6.30 RfidKillTagEx

killing the UHF RFID tag which in order to not response to the command of reader.

If the kill's password is "00000000", then cannot perform kill. In order to perform kill command, tag must be set with other password, not "00000000". Once tag be killed, it never recoveragain.

Support MASK function of bit units.

```
RFID_RESULTRfidKillTagEx (
    BOOLbSyncMode,
    LPWSTRszKillPassword,
    BOOLbContinuous,
    RFIDPMASK_PARAMS_EXpMask,
);
```

Parameters

bSyncMode

mode of operation of function. (True=Sync, False=Async)

szKillPassword

Kill Password saved in Tag

bContinuous

Continuous mode. (True=Enable, False=Disable)

Mask

masking pattern of bit units..

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully..

Notes

Continuous mode is not supported in Sync mode.

Once performed this function that will never be get restored again.

2.6.31 RfidKillTagEx2

killing the UHF RFID tag which in order to not response to the command of reader.

If the kill's password is "00000000", then cannot perform kill. In order to perform kill command, tag must be set with other password, not "00000000". Once tag be killed, it never recoveragain.

Support SELECT function and MASK function of bit units.

```
RFID_RESULT RfidKillTagEx2 (
    BOOL bSyncMode,
```

```
    LPWSTR szKillPassword,
    BOOL bContinuous,
    PRFID_MASK_PARAMS_EX2 pMasks,
);
```

Parameters

bSyncMode

mode of operation of function. (True=Sync, False=Async)

szKillPassword

Kill Password saved in Tag

bContinuous

Continuous mode. (True=Enable, False=Disable)

Mask

Address of masking pattern of bit units..

nNumberOfMasks

The number of data of pMasks.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully..

Notes

If you use pMasks, Sync mode is not supported.

Once performed this function that will never be get restored again.

It can be supported, if module type is MODULE_TYPE_H1000

2.6.32 RfidSetDefault

Initialize the UHF RFID reader module values.

```
RFID_RESULTRfidSetDefault ();
```

Parameters

None

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.33 RfidSetExtInfoEnable

Enable or disable to read RSSI value additionally while reading Tag.

```
RFID_RESULTRfidSetExtInfoEnable (
    BOOLbEnable
);
```

Parameters

bEnable

TRUE: set Extended Information Mode.

FALSE: remove setting of Extended Information Mode.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.34 RfidGetQValue

Get the Q Parameter value that is used at ReadEpc.

```
RFID_RESULT    RfidGetQValue (
        UINT* pnQvalue
);
```

Parameters

pnQvalue

value of Q parameter. (Min:0 ~ Max:15)

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.35 RfidSetQValue

Set the Q Parameter value that is used at ReadEpc.

```
RFID_RESULT    RfidSetQValue (
        UINT nQvalue
);
```

Parameters

nQvalue

Q Parameter (Min:0 ~ Max:15)

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.36 RfidGetOperationTime

Get the execution time of RFID device command. The unit as sec. and if it be 0, the command executed completely or stand by till get Stop command from user.

```
RFID_RESULT    RfidGetOperationTime (
        UINT* pnOperationTime
);
```

Parameters

pnOperationTime

operation time. (Min:0 ~ Max:60)

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.37 RfidSetOperationTime

Set the execution time of RFID command. The unit as sec. and if it be 0, the command executed completely or stand by till get Stop command from user.

```
RFID_RESULT    RfidSetOperation (
        UINT nOperationTime
);
```

Parameters

nOperationTime

Operation Time(second)(Min:0 ~ Max:60)

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.38 RfidGetPowerLevel

Get RFID's Power Level.

```
RFID_RESULT    RfidGetPowerLevel (
        UINT* pnPowerLevel
);
```

Parameters

pnPowerLevel

power level (Min:0 ~ Max:31) default value(about 28dBm)

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.39 RfidSetPowerLevel

Set RFID device's Power Level.

```
RFID_RESULT    RfidSetPowerLevel (
        UINT nPowerLevel
);
```

Parameters

nPowerLevel

Power level (Min:0 ~ Max:31) default value(about 28dBm)

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.40 RfidGetSession

Get the Session value that will be used at ReadEpc

```
RFID_RESULT    RfidGetSession (
        RFIDSESSIONTYPE* pSessionType
);
```

Parameters

pSessionType

Session

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.41 RfidSetSession

Set the Session value that will be used at ReadEpc.

```
RFID_RESULT    RfidSetSession (
        RFIDSESSIONTYPE SessionType
);
```

Parameters

SessionType

Session

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.42 RfidGetInventoryTarget

Get the target while performing the ReadEpc.

```
RFID_RESULT    RfidGetInventoryTarget (
        RFIDINVENTORIEDSTATE* pInventoriedState
);
```

Parameters

pInventoriedState

Inventory Target

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.43 RfidSetInventoryTarget

Set the target while performing the ReadEpc.

```
RFID_RESULT    RfidSetInventoryTarget (
        RFIDINVENTORIEDSTATE InventoriedState
);
```

Parameters

InventoriedState

Inventory Target

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.44 RfidGetLbtChState

Get LBT Channel by user (Apply to Japan, Europe type)

the frequency of use and the number of channels of Each Module is shown in the table below.

MODULE_TYPE_H1000		MODULE_TYPE_I2000	
Europe	Japan	Japan 1W	Japan 250mW
865.7	X	916.8	916.8
866.3	952.4	918.0	918.0
866.9	952.6	919.2	919.2

867.5	952.8	920.4	920.4
	953.0	920.6	920.6
	953.2	920.8	920.8
	953.4		921.0
	953.6		921.2
	953.8		921.4
	X		921.6
			921.8
			922.0
			922.2
			922.4
			922.6
			922.8
			923.0
			923.2
			923.4

```
RFID_RESULT    RfidGetLbtChState (
        UINT* pnLbtChState
);
```

Parameters

pnLbtChState

LBT channel state Min:0(0x00) ~ Max:524287(0x7FFFF)

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

When change this value to binary numeral, the least significant bit will become Channel1 and if the bit is 1, use that channel.

2.6.45 RfidSetLbtChState

Set LBT Channel by user (Apply to Japan, Europe type)

Reference to 2.6.44 RfidGetLbtChState table

```
RFID_RESULT    RfidSetLbtChState (
        UINT nLbtChState
);
```

Parameters

nLbtChState

LBT channel stateMin:0(0x00) ~ Max:524287(0x7FFFF)

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

When change this value to binary numeral, the least significant bit will become Channel1 and if the bit is 1, use that channel.

2.6.46 RfidGetChannelState

You can use RfidGetChannelState function instead of RfidGetLbtChState to get the channel state, if module type is MODULE_TYPE_H1000 and Europe.

```
RFID_RESULT    RfidGetChannelState (
        UINT* pnChState
);
```

Parameters

pnChState

Channel State

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

When change this value to binary numeral, the least significant bit will become Channel1 and if the bit is 1, use that channel.

2.6.47 RfidSetChannelState

You can use RfidGetChannelState function instead of RfidGetLbtChState to set the channel, if module type is MODULE_TYPE_H1000 and Europe.

```
RFID_RESULT    RfidSetChannelState (
        UINTnChState
);
```

Parameters

nChState

Channel State

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

When change this value to binary numeral, the least significant bit will become Channel1 and if the bit is 1, use that channel.

2.6.48 RfidGetLbtTime

Get the time, which to occupy a LBT channel and to read Tag. (apply to Japan, Europe Type)And the units is ms and according to the radio regulations the occupy time cannot exceed 4 seconds.

```
RFID_RESULT    RfidGetLbtTime (
        UINT* pnLbtTime
);
```

Parameters

pnLbtTime

LBT Time (Min:0 ~ Max:4000)

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

It can be supported if module type is MODULE_TYPE_H1000

2.6.49 RfidSetLbtTime

Set the time, which to occupy a channel and to read Tag. (apply to Japan, Europe Type)

And the units is ms and according to the radio regulations the occupy time cannot exceed 4 seconds.

```
RFID_RESULT    RfidSetLbtTime (
        UINT nLbtTime
);
```

Parameters

nLbtTime

LBT Time (Min:0 ~ Max:4000)

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.50 RfidSetAirDuty

Set Tx cycle of RFID device.

```
RFID_RESULT    RfidSetAirDuty (
        UINT nSpeed
);
```

Parameters

nSpeed

Speed value. (Min:10 ~ Max:100)

As the value increases, the speed increases.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

Don't use this function for general purpose.because the optimal value is aleady set about battery consumption and heat generation.

It can be supported if module type is MODULE_TYPE_H1000

2.6.51 RfidGetFirmwareVersion

Read the firmware version of RFID Module.

```
RFID_RESULT    RfidGetFirmwareVersion (
        LPWSTR szVersion
);
```

Parameters

szVersion

firmware version

Return Values.

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.52 RfidGetModuleType

Return type of RFID module equipped with PDA.

```
uint    RfidGetModuleType ( );
```

Parameters

None

Return Values

There are some differences according to the type of RFID module.

Reference to 2.2 Constants

2.6.53 RfidGetXcvrMaxPwr

Return the maximum Tx Power value of RFID module equipped with PDA.

```
RFID_RESULT   RfidGetXcvrMaxPwr (
  UINT * pPwr
);
```

Parameters

pPwr

30x : 1W

28x : 600mW

27x : 500mW

24x : 250mW

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

2.6.54 RfidIsRunning

Return whether operation of RFID moduel is in progress.

```
BOOL   IsRunning ( );
```

Parameters

None

Return Values

It can be supported if module type is MODULE_TYPE_H1000

If module operation is in progress, TRUE will be returned, and if not, False will be returned

2.6.55 RfidGetR1000MacReg

Read register value of R1000

```
BOOLRfidGetR1000MacReg (
```

```
    unsigned shortReg,
    unsigned int *pValue
);
```

Parameters

Reg

Register address(reference to a separatedocument)

pValue

Variable which register value will be stored in

Return Values

True will be returned if performed successfully.

Notes

It can be supported if module type is MODULE_TYPE_H1000

2.6.56 RfidSetR1000MacReg

Write value into register of R1000.

```
BOOLRfidSetR1000MacReg (
    unsigned shortReg,
    unsigned intValue
);
```

Parameters

Reg

Register address(reference to a separatedocument)

Value

Variable which register value will be stored in

Return Values

True will be returned if performed successfully.

Notes

It can be supported if module type is MODULE_TYPE_H1000

2.6.57 RfidSetTagFocus

Enable/Disable Impinj Extension(Tag Focus) function.

InventoryTarget, Session properties cannot be changed, while EnableTagFocus property is set to true.

```
RFID_RESULT    RfidSetTagFocus (
```

```
    BOOL bEnable,
);
```

Parameters

bEnable

Enable/Disable state of Tag Focus function.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

Only in support of MODULE_TYPE_I2000

2.6.58 RfidGetTagFocusStatus

Read Enable/Disable state of Impinj Extension(Tag Focus) function.

InventoryTarget, Session properties cannot be changed, while EnableTagFocus property is set to true.

```
RFID_RESULT   RfidGetTagFocusStatus (
    BOOL * pbEnable,
);
```

Parameters

pbEnable

variable where Enable/Disable state of Tag Focus function will be stored in.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

Only in support of MODULE_TYPE_I2000

2.6.59 RfidSetAirDura

Set the operating time of Tx cycle.

```
RFID_RESULT   RfidSetAirDura (
    RFIDAIRDURAPARAMS * pRfidParams
);
```

Parameters

pRfidParams

Tx Cycle의 on시간과 Off시간이저장되어있는구조체의포인터

The pointer which On time and Off time of Tx Cycle are stored in.

Return Values

RFID_RESULT_SUCCESS will be returned if performed successfully.

Notes

If MODULE_TYPE_I2000, Tx Cycle operating time default is On(200ms), Off(0ms).

2.6.60 RfidGetEuroMode

Read the operating way of the module, if module type MODULE_TYPE_H1000 and Europe

```
BOOL    RfidGetEuroMode (
        Unsigned int * pmode
);
```

Parameters

mode

1 : four channels hopping

2 : one channel of 4 channels is selected by the random and fixed.

Return Values

RFID_RESULT_SUCCESS will be returned, if performed successfully.

Notes

2.6.61 RfidSetEuroMode

set the operating way of the module, if module type MODULE_TYPE_H1000 and Europe

```
BOOL    RfidSetEuroMode (
        Unsigned int mode
);
```

Parameters

mode

1 : four channels hopping

2 : one channel of 4 channels is selected by the random and fixed.

Return Values

RFID_RESULT_SUCCESS will be returned, if performed successfully.

Notes